

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2000020481 A**

(43) Date of publication of application: **21.01.00**

(51) Int. Cl.

**G06F 15/16**  
**G06T 1/20**

(21) Application number: **10190096**

(22) Date of filing: **06.07.98**

(71) Applicant: **HITACHI LTD**

(72) Inventor: **YOSHIDA MASASHI**  
**IKEDA KOJI**  
**TAKANE ATSUSHI**  
**SATO NORIO**

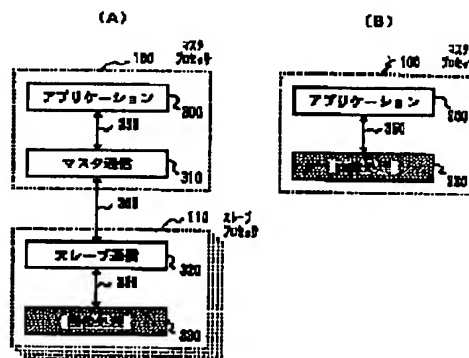
**(54) PARALLEL IMAGE PROCESSING SYSTEM**

**(57) Abstract:**

**PROBLEM TO BE SOLVED:** To attain the miniaturization/cost-down of a device by providing the device through minimum processes without performing recompiling or the like so as to execute a program, which is operated between parallel processors, through a single processor.

**SOLUTION:** An interface for calling a master communication program from an application 300 is made equal with an interface for calling an image processing program from a slave communication program, the image processing program is directly called from the application 300 and the image processing program can be executed on a master.

COPYRIGHT: (C)2000,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-20481

(P2000-20481A)

(43) 公開日 平成12年1月21日 (2000.1.21)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード* (参考)
G 0 6 F 15/16	3 8 0	G 0 6 F 15/16	3 8 0 Z 5 B 0 4 5
G 0 6 T 1/20		15/66	K 5 B 0 5 7

審査請求 未請求 請求項の数 1 O L (全 9 頁)

(21) 出願番号 特願平10-190096

(22) 出願日 平成10年7月6日 (1998.7.6)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 吉田 昌司

茨城県日立市大みか町七丁目1番1号 株式会社日立製作所日立研究所内

(72) 発明者 池田 光二

茨城県日立市大みか町七丁目1番1号 株式会社日立製作所日立研究所内

(74) 代理人 100068504

弁理士 小川 勝男

最終頁に続く

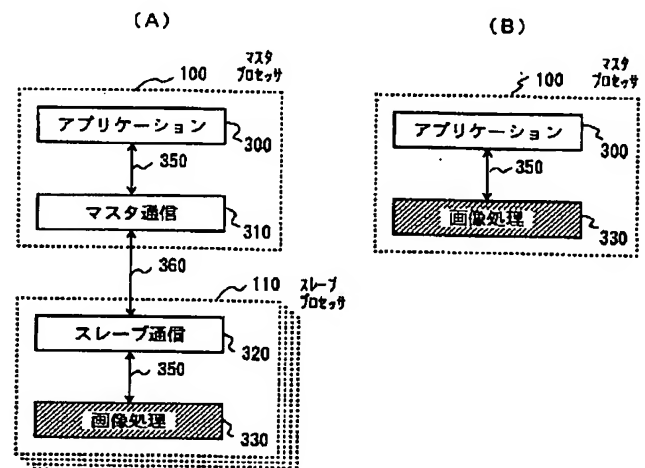
(54) 【発明の名称】 並列画像処理システム

(57) 【要約】

【課題】 並列プロセッサで動作するプログラムを、単一プロセッサで実行するように、再コンパイルなどを行うことなく、最小の工数で実現し、装置の小型化/装置コストの削減を図る。

【解決手段】 アプリケーションからマスタ通信プログラムを呼び出すインタフェースと、スレーブ通信プログラムから画像処理プログラムを呼び出すインタフェースを同一にし、アプリケーションから画像処理プログラムを直接呼び出して、画像処理プログラムをマスタ上で実行することを可能とする。

図 3



## 【特許請求の範囲】

## 【請求項 1】 マスタプロセッサと、

前記マスタプロセッサと同一の命令で動く複数のスレーブプロセッサとからなる並列画像処理システムにおいて、

前記マスタプロセッサ上で動作する処理部と、

前記マスタプロセッサ上で動作し、画像をそれぞれの前記スレーブプロセッサ用に分割し、それぞれの前記スレーブプロセッサとの間でデータ転送を行うマスタ通信部と、

前記スレーブプロセッサ上で動作し、前記マスタプロセッサとのデータ転送を行うスレーブ通信部と、

前記スレーブプロセッサ上で動作し、分割された画像に対して画像処理をする画像処理部とを有し、

前記処理部と前記マスタ通信部間のデータ転送手段が、前記スレーブ通信部と前記画像処理部間のデータ転送手段と同一である並列画像処理システム。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】 本発明は、データ処理計算機の技術分野に属し、マスタ／スレーブ型並列プロセッサ画像処理装置上のプログラムの構成に關している。

## 【0002】

【従来の技術】 複数のプロセッサを用いて処理を行う並列プロセッサは、性能向上を図る技術として、かなり以前より研究／開発されてきた。また、画像処理は画素毎に独立な処理となりやすく、処理の並列性が高い性質があるため、他の分野と比較して並列プロセッサにより大きく性能を向上でき、並列プロセッサを用いた画像処理装置も数多く研究／開発されてきている。

【0003】 従来の並列プロセッサ技術は、プログラムを作る工数を減らし、またなるべく多くの分野で用いられるような汎用性を持たせるために、単一プロセッサ用のプログラムを並列プロセッサ用のプログラムに自動変換する、並列コンパイラという技術がある。並列コンパイラに關した文献／特許は、“佐藤誠他、「パターン情報処理用マルチマイクロプロセッサシステム P X-1」、信学論 Vol. 64, No. 11, p. 1021-1028 (Nov., 1981) ”、“Robert G. Babb, 「並列処理マシン用 Fortran の H P F、業界標準の地位を目指す」、日経エレクトロニクス, no. 581 (1993. 5. 24) ”、特開平 9-91258 号、特開平 8-328871 号、特開平 7-311746 号、特開平 6-26481 号などがある。

【0004】 また、並列プロセッサを実用化するためには、データを各プロセッサにどのように振り分けるかということが重要であり、このデータ分割の方法に關した特許としては、特開平 10-69469 号、特開平 8-278949 号、特開平 6-325162 号などが挙げられる。

【0005】 さらに、その他並列プロセッサの制御や装置に關した文献／特許としては、“小西弘一、「並列機

用インターフェース M P I、アプリ開発支援機能を盛り込む、メッセージ通信の業界標準に」日経エレクトロニクス, no. 605 (1994. 4. 11) ”、特開平 8-249018 号、特開平 8-249022 号、特開平 8-249018 号、特開平 8-161271 号、特開平 8-16754 号、特開平 7-51594 号、特開平 5-257934 号、特開平 5-233569 号、米国特許 5050070、その他数々の文献／特許がある。

## 【0006】

【発明が解決しようとする課題】 最近の汎用プロセッサの性能向上はめざましく、並列プロセッサでなければ実現できなかった性能が、短期間のうちに単一プロセッサで達成される状況にある。従って、並列プロセッサを簡単に単一プロセッサへ変更できれば、装置の小型化がはかれ、コストも安くてすむ。従って、並列プロセッサとして開発したプログラムを単一プロセッサで実行する形式へ簡単に変更できる方法が課題となる。並列コンパイラという従来技術は、並列プロセッサとして開発したプログラムを単一プロセッサで実行する形式にコンパイルし直せばよいので、並列プロセッサとして開発したプログラムを単一プロセッサで実行する形式に変更することが可能である。しかし、プログラムを作る工数は減るが、並列コンパイラを作るための工数がかかり、なるべく多くの分野で用いられるような汎用性をねらわずに画像処理のみに適用するのであれば、むしろ並列コンパイラを作るための工数が大きいことが問題となる。

【0007】 また、並列コンパイラ以外で上に挙げた従来技術においては、並列プロセッサの並列性を高め性能を向上させるという観点の技術しかなく、並列プロセッサとして開発したプログラムを単一プロセッサで実行させるという観点の技術が述べられた例はない。

【0008】 本発明の課題は、並列プロセッサで動作するプログラムを単一プロセッサでも実行できるような仕組みを、再コンパイルなどを行うことなく、最小の工数で実現することにある。

## 【0009】

【課題を解決するための手段】 上記課題を解決するために、本発明は、マスタプロセッサと、前記マスタプロセッサと同一の命令で動く複数のスレーブプロセッサとからなる並列画像処理システムにおいて、前記マスタプロセッサ上で動作する処理部と、前記マスタプロセッサ上で動作し、画像をそれぞれの前記スレーブプロセッサ用に分割し、それぞれの前記スレーブプロセッサとの間でデータ転送を行うマスタ通信部と、前記スレーブプロセッサ上で動作し、前記マスタプロセッサとのデータ転送を行うスレーブ通信部と、前記スレーブプロセッサ上で動作し、分割された画像に対して画像処理をする画像処理部とを有し、前記処理部と前記マスタ通信部間のデータ転送手段が、前記スレーブ通信部と前記画像処理部間のデータ転送手段と同一とした。

## 【0010】

【発明の実施の形態】図1は、マスタ/スレーブ型並列プロセッサのハードウェア構成例である。

【0011】マスタプロセッサ100、グローバルメモリ101、マスタバス102、スレーブプロセッサ110、ローカルメモリ111、スレーブバス112、バススイッチ113、スレーブプロセッサ120、ローカルメモリ121、スレーブバス122、バススイッチ123、スレーブプロセッサ130、ローカルメモリ131、スレーブバス132、バススイッチ133、スレーブプロセッサ140、ローカルメモリ141、スレーブバス142、バススイッチ143から成る。グローバルメモリ101は、1画像全体の画像データを保持しており、マスタプロセッサ100は、その画像データをスレーブプロセッサの数である4つに分割し、ローカルメモリ111、121、131、141へ転送するが、例えばローカルメモリ121へ送る場合は、バススイッチ123をONにし、それ以外のバススイッチ113、133、143をOFFとし、マスタバス102、バススイッチ123、スレーブバス122を通じてローカルメモリ121へ転送する。このとき、例えばバススイッチ113はOFFになっているので、スレーブプロセッサ110は、スレーブバス112を通じて、ローカルメモリ111から分割した画像を読みだし、画像処理し、結果をローカルメモリ111に書き込むことができる。マスタプロセッサ100が他のローカルメモリ111、131、141へ転送する際も同様である。また、本例は、スレーブプロセッサが4つの例を示したが、数を増した場合及び数を減らした場合も全く同様に構成することができる。図2は、マスタのみの単一プロセッサのハードウェアに変更した時のハードウェア部分の構成例である。図1のマスタ/スレーブ型並列プロセッサから、スレーブ110、120、130、140及びそれに伴うローカルメモリ111、121、131、141、バススイッチ113、123、133、143を取り去った形になっている。このように、マスタ/スレーブ型並列プロセッサのハードウェアは、スレーブとその周辺回路を取り除くだけで、単一プロセッサハードウェアに容易に変更可能である。

【0012】図3は、ソフトウェアの構成例である。

(A)は、図1のマスタ/スレーブ型並列プロセッサのハードウェアに対応する図、(B)は、図2の単一プロセッサのハードウェアに対応する図である。(A)は、アプリケーション300、マスタ通信310、スレーブ通信320、画像処理330からなる。アプリケーション300、マスタ通信310は図1におけるマスタプロセッサ100上で動作するプログラムであり、スレーブ通信320、画像処理330は、図1におけるスレーブプロセッサ、たとえばスレーブプロセッサ110上で動作するプログラムである。これは、残りのスレーブプロセッサ120、130、140でも全く同じ構成とな

る。アプリケーション300はインタフェース350を用いてマスタ通信310を呼び出し、データをやり取りする。マスタ通信310はマスタスレーブインタフェース360を用いてスレーブ通信320を呼び出し、データをやり取りする。スレーブ通信320は、アプリケーション300がマスタ通信310とデータをやり取りするインタフェースと全く同一のインタフェース350を用いて画像処理330を呼び出し、データをやり取りする。(B)はアプリケーション300、画像処理330からなり、両者は図2のマスタプロセッサ100上で動作するプログラムである。アプリケーション300はインタフェース350を用いて画像処理330を呼び出し、データをやり取りする。このように、(A)において、アプリケーション300からマスタ通信310を呼び出すインタフェース350と、スレーブ通信320から画像処理330を呼び出すインタフェース350を同一にしておくことにより、(B)のようにアプリケーション300から画像処理330を直接呼び出すことができ、マスタスレーブ型マルチプロセッサの装置からスレーブを取り去ってマスタのみの単一プロセッサの装置に変更することが簡単に可能となる。

【0013】図4は、呼び出し関係を説明する図である。(A)は、アプリケーション300からインタフェース350を用いてマスタ通信プログラム310を呼び出し、データをやり取りする例である。(B)は、スレーブ通信320からインタフェース350を用いて画像処理330を呼び出し、データをやり取りする例である。アプリケーション300にはマスタ通信プログラム310を呼び出す関数を書いてあり、引数データは一つの構造体のインタフェース350にまとめ、そのアドレスを渡す。引数データを渡されたマスタ通信プログラム310は、処理を行い、結果を同じインタフェース350にまとめて返す。スレーブ通信320は、同じように画像処理330を呼び出す関数を書いてあり、引数データは同じ形の一つの構造体のインタフェース350にまとめ、そのアドレスを渡す。引数データを渡された画像処理330は、処理を行い、結果を同じように同じインタフェース350にまとめて返す。

【0014】図5は、インタフェース350の構成例である。(A)はインタフェース350の構成例、(B)はインタフェース350の中で使われる画像構造体PTN\_MCBの構成例、(C)はインタフェース350の中で使われるウィンドウ構造体PTN\_WCBの構成例である。インタフェース350は、入力画像構造体ImgSrc501、入力ウィンドウ構造体WinSrc502、出力画像構造体ImgDst503、出力ウィンドウ構造体WinDst504、フィルタカーネルの横幅KerH505、フィルタカーネルの縦幅KerW506からなる。画像構造体PTN\_MCBは、画像に関するパラメータを一つの構造体にしたもので、画像の先頭アドレスAddress511、画像の論理的横幅Width512、

画像の論理的縦幅Height513, 画像の物理的実体の横幅 a\_Width514, 画像の物理的実体の縦幅 a\_Height515, 一画素を表すのに用いられるビット数depth516からなる。ここで、論理的幅とっているのは、実際に画素のデータが入っている幅で、物理的実体の幅というのは、アクセスに便利のように論理的幅より大きい16の倍数にして実際にメモリ上に割り当てる幅のことである。ウィンドウ構造体PTN\_WCBは、画像上におけるウィンドウの先頭x座 x\_coord521, 画像上におけるウィンドウの先頭y座標 y\_coord522, ウィンドウの横幅Width 523, ウィンドウの縦幅Height524 からなる。このように、インタフェース350は画像のパラメータなどからなっており、一つの画像を複数のスレーブに分割した場合、画像の大きさは変わるが、パラメータの構造自体は同じにすることができる。

【0015】図6は、マスタスレーブ間のインタフェース360の例である。マスタスレーブ間インタフェース360は、マスタ通信310とスレーブ通信320を結ぶものである。下側にむかうにつれて処理が進む形で記述してある。マスタスレーブ間インタフェース360は、ローカルメモリ（例えば110）上にデータを置くことでやりとりを実現する。ローカルメモリ上のデータには、ローカルメモリのアクセス権631~635, 画像データ641~647からなる。ローカルメモリのアクセス権は1ビットで、0がマスタにアクセス権あり、1がスレーブにアクセス権ありを示す。マスタ通信310はまず、画像をスレーブの数に分割する画像分割611を行い、その後にローカルメモリ上に領域を設けるメモリ確保612を行う。このとき、アクセス権631と画像データ641の領域が設けられ、アクセス権は0に初期化され、すなわちマスタにアクセス権ありがデフォルトとなる。次に最初の1ラインの画像データを送る画像データ送信613が行われ、画像データ642にデータが送られる。データが送り終わるとアクセス権を1にする処理614が行われ、アクセス権632は1の値となる。マスタ通信310はスレーブ起動615を実行し、スレーブを立ち上げる。612から615の処理はスレーブの数だけ行う必要があり、ループ判定616にて、スレーブ数nの回数だけ繰り返す。スレーブ上では、スレーブ通信処理320を介し、画像処理330が実行開始する。その後、マスタ通信310はアクセス権が0にされるのを待つ処理617によって、待つ。画像処理330は、画像データ643を読み出し、演算661を行って、結果を画像データ643に書き込む。一行分が終わったら、最終行か否かの判定662を行い、最終行でない場合は、スレーブ通信処理320がアクセス権をマスタに返す処理651を行い、マスタがアクセス権をスレーブに渡してくるまで待つ処理652に入る。アクセス権が0にされるのを待つ処理617を行っていたマスタ通信310はアクセス権0を検出すると画像デ

ータ644を読み込む画像データ受信618を行い、1行分の結果を集めたあと、画像データ送信619により次の1行分のデータ645を送る。1行分送り終わったら、アクセス権634を1にする処理620を行い、これをスレーブ数ずつ繰り返す処理621とさらに画像の縦幅分繰り返す処理622により、処理を継続する。マスタがアクセス権をスレーブに渡してくるまで待つ処理652に入っていたスレーブ通信320は、アクセス権634がマスタ通信310によって1にされると、画像処理330に次の行に関する演算661を行わせる。こうして画像全体の処理を進め、最後の行において、最終行か否かの判定662で最後の行であると判定されると、スレーブ通信320はアクセス権635を0にする処理653を行って処理を終了する。マスタ通信310もこのときアクセス権635が0かどうかを判定する処理623で待っているはずで、アクセス権635が0であると判定すると最後の行の演算結果である画像データ646を集める画像データ受信624を行い、これをスレーブの数だけ繰り返す処理625で繰り返した後、処理を終了する。このように、マスタ通信310とスレーブ通信320は、インタフェース360を介して、アクセス権631~635で一行ずつ同期をとりながら画像データ641~646をやりとりして処理を進めていく。

【0016】図7は、画面を縦に分割する場合の入力画像の構成例を示したものである。マスタ通信310で画像をスレーブの数SlaveNumに分割する画像分割611での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、入力画像をスレーブの数に分けるとときには、境界部分でカーネルの大きさ分だけ重なり合わせた形で分けなければならない。スレーブの入力画像Slv->ImgSrcは、マスタの入力画像Mst->ImgSrcのパラメータを用いて次のような式で表される。

【0017】
$$\begin{aligned} \text{Slv} \rightarrow \text{ImgSrc.Width} &= \{ \text{Mst} \rightarrow \text{WinSrc.Width} - \\ &\quad (\text{Mst} \rightarrow \text{KerW} - 1) \} / \text{SlaveNum} + (\text{Mst} \rightarrow \text{KerW} - 1) \\ \text{Slv} \rightarrow \text{ImgSrc.Height} &= \text{Mst} \rightarrow \text{WinSrc.Height} \\ \text{Slv} \rightarrow \text{WinSrc.x\_coord} &= 0, \text{Slv} \rightarrow \text{WinSrc.y\_coord} = 0 \\ \text{Slv} \rightarrow \text{WinSrc.Width} &= \{ \text{Mst} \rightarrow \text{WinSrc.Width} - (\text{Mst} \rightarrow \text{KerW} - 1) \} / \text{SlaveNum} + (\text{Mst} \rightarrow \text{KerW} - 1) \\ \text{Slv} \rightarrow \text{WinSrc.Height} &= \text{Mst} \rightarrow \text{WinSrc.Height} \end{aligned}$$

図8は、画面を縦に分割する場合の出力画像の構成例を示したものである。マスタ通信310で画像をスレーブの数SlaveNumに分割する画像分割611での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、出力画像は周囲部分でカーネルの大きさの半分の分だけ計算できない領域が発生する。図ではこの領域をハッチングして表している。スレーブの出力画像Slv->ImgDstは、マスタの入力画像Mst->ImgDstのパラメータを用い

て次のような式で表される。

```

【0018】 Slv->ImgDst.Width={Mst->WinDst.Width-
(Mst->KerW-1)}/SlaveNum+(Mst->KerW-1)
Slv->ImgDst.Height=Mst->WinDst.Height
Slv->WinDst.x_coord=0, Slv->WinDst.y_coord=0
Slv->WinDst.Width={Mst->WinDst.Width-(Mst->KerW-
1)}/SlaveNum
Slv->WinDst.Height=Mst->WinDst.Height

```

図9は、画面を横に分割する場合の入力画像の構成例を示したものである。マスタ通信310で画像をスレーブの数SlaveNumに分割する画像分割611での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、入力画像をスレーブの数に分けるとときには、境界部分でカーネルの大きさ分だけ重なり合わせた形で分けなければならない。スレーブの入力画像Slv->ImgSrcは、マスタの入力画像Mst->ImgSrcのパラメータを用いて次のような式で表される。

```

【0019】 Slv->ImgSrc.Height={Mst->WinSrc.Height-
(Mst->KerH-1)}/SlaveNum+(Mst->KerH-1)
Slv->ImgSrc.Width=Mst->WinSrc.Width
Slv->WinSrc.x_coord=0, Slv->WinSrc.y_coord=0
Slv->WinSrc.Height={Mst->WinSrc.Height-(Mst->KerH-
1)}/SlaveNum+(Mst->KerH-1)
Slv->WinSrc.Width=Mst->WinSrc.Width

```

図10は、画面を横に分割する場合の出力画像の構成例を示したものである。マスタ通信310で画像をスレーブの数SlaveNumに分割する画像分割611での処理を示している。一般に画像処理は、注目画素の周りのカーネルの大きさの領域を読み込んで演算するので、出力画像は周囲部分でカーネルの大きさの半分の分だけ計算できない領域が発生する。図ではこの領域をハッチングして表している。スレーブの出力画像Slv->ImgDstは、マスタの入力画像Mst->ImgDstのパラメータを用いて次のような式で表される。

```

【0020】 Slv->ImgDst.Height={Mst->WinDst.Height-
(Mst->KerH-1)}/SlaveNum+(Mst->KerH-1)
Slv->ImgDst.Width=Mst->WinDst.Width
Slv->WinDst.x_coord=0, Slv->WinDst.y_coord=0
Slv->WinDst.Height={Mst->WinDst.Height-(Mst->KerH-
1)}/SlaveNum
Slv->WinDst.Width=Mst->WinDst.Width

```

図11は、第2のソフトウェア構成例である。アプリケーション300、マスタ管理1100、マスタ通信310、スレーブ通信320、画像処理330からなる。アプリケーション300、マスタ管理1100、マスタ通信310、画像処理330は図1におけるマスタプロセッサ100上で動作するプログラムであり、スレーブ通信320、画像処理330は、図1におけるスレーブ

ロセッサ、たとえばスレーブプロセッサ110上で動作するプログラムである。これは、残りのスレーブプロセッサ120、130、140でも全く同じ構成となる。アプリケーション300は、インタフェース1110を用いてマスタ管理1100を呼び出し、データをやり取りする。マスタ管理1100は、インタフェース350を用いてマスタ通信310または画像処理330のどちらかを呼び出し、データをやり取りする。マスタ通信310はマスタスレーブインタフェース360を用いてスレーブ通信320を呼び出し、データをやり取りする。スレーブ通信320は、アプリケーション300がマスタ通信310とデータをやり取りするインタフェースと全く同一のインタフェース350を用いて画像処理330を呼び出し、データをやり取りする。このように、マスタ管理1100からマスタ通信310または画像処理330を呼び出すインタフェース350と、スレーブ通信320から画像処理330を呼び出すインタフェース350を同一にしておくことにより、画像処理330をマスタで実行させることも、スレーブで実行させることも可能となる。

#### 【0021】

【発明の効果】本発明によれば、並列プロセッサで動作するプログラムを、単一プロセッサで実行するように、再コンパイルなどを行うことなく、最小の工数で実現でき、並列プロセッサの装置を、単一プロセッサの装置に変更できるため、装置の小型化／装置コストの削減という効果がある。

#### 【図面の簡単な説明】

【図1】本発明からなる画像処理システムの一実施例を示す図である。

【図2】本発明からなる画像処理システムの他の実施例を示す図である。

【図3】本発明からなる画像処理システムのソフトウェア部に関する一実施例を示す図である。

【図4】本発明からなる画像処理システムのデータ通信を説明する図である。

【図5】本発明からなる画像処理システムのインタフェースの一実施例を示す図である。

【図6】本発明のインタフェースのフローチャートを示す図である。

【図7】本発明の画面を縦に分割する場合の入力画像の一構成例を示す図である。

【図8】本発明の画面を縦に分割する場合の出力画像の一構成例を示す図である。

【図9】本発明の画面を横に分割場合の入力画像の一構成例を示す図である。

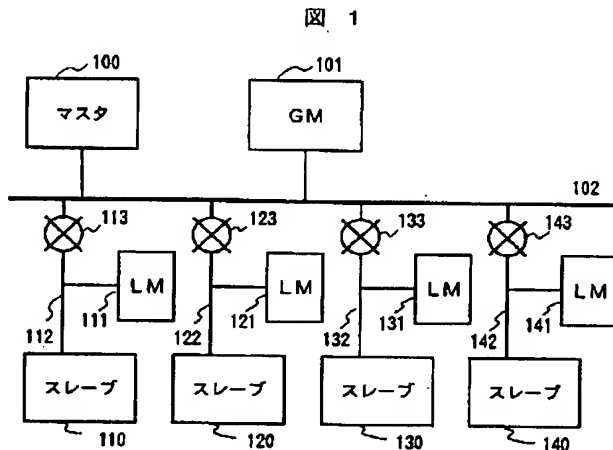
【図10】本発明の画面を横に分割場合の出力画像の一構成例を示す図である。

【図11】本発明からなる画像処理システムのソフトウェア部に関する他の実施例を示す図である。

## 【符号の説明】

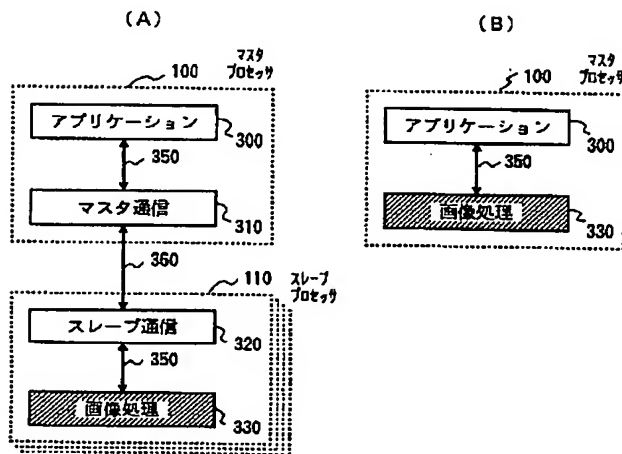
100…マスタプロセッサ、101…グローバルメモリ、102…マスタバス、110, 120, 130, 140…スレーブプロセッサ、111, 121, 131, 141…ローカルメモリ、112, 122, 132, 142…スレーブバス、113, 123, 133, 143…バススイッチ、300…アプリケーション、310…マスタ通信、320…スレーブ通信、330…画像処理、350…インタフェース、360…マスタスレーブインタフェース、501…入力画像構造体、502…入力ウィンドウ構造体、503…出力画像構造体、504…出力ウィンドウ構造体、505…フィルタカーネルの横幅、506…フィルタカーネルの縦幅、511…先頭アドレス、512…論理値横幅、513…論理値縦幅、514…物理的実体の横幅、515…物理的実体の縦幅、516…ビット数、521…ウィンドウの先頭X座標、522…ウィンドウの先頭Y座標、523…ウィンドウの横幅、524…ウィンドウの縦幅、611…画像分割、612…メモリ確保、613, 619…画像データ送信、614, 620…アクセス権を1にする処理、615…スレーブ起動、616…ループ判定、617…アクセス権が0にされるのを待つ処理、618, 624…画像データ受信、621, 625…繰り返す処理、622…縦幅分繰り返す処理、623…アクセス権が0かどうか判定する処理、631, 632, 633, 634, 635…アクセス権、641, 642, 643, 644, 646…画像データ、645…1行分のデータ、651…アクセス権をマスタに返す処理、652…アクセス権をスレーブに渡してくれるまで待つ処理、653…アクセス権を0にする処理、661…演算、662…最終行か否かの判定、1100…マスタ管理、1110…インタフェース。

【図1】



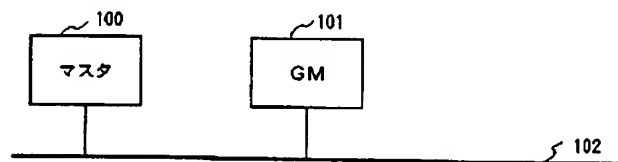
【図3】

図 3

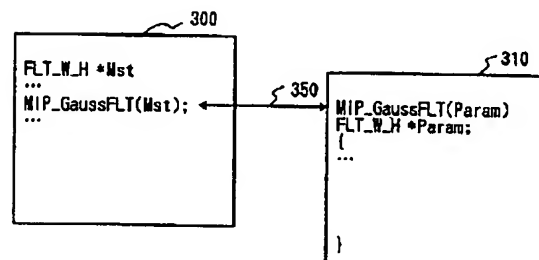


【図2】

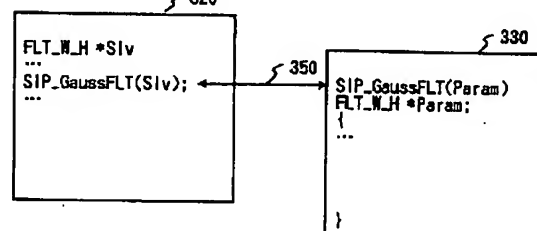
図 2



【図4】

図 4  
(A)

(B)



【図 5】

図 5

(A)

```

typedef struct(
    PTN_MCB      ImgSrc;    ~501    /* 入力画像 */
    PTN_WCB      WinSrc;    ~502    /* 入力ウィンドウ */
    PTN_MCB      ImgDst;    ~503    /* 出力画像 */
    PTN_WCB      WinDst;    ~504    /* 出力ウィンドウ */
    unsigned short KerH;    ~505    /* カーネル縦幅 */
    unsigned short KerW;    ~506    /* カーネル縦幅 */
) FLT_WLH;

```

(B)

```

typedef struct(
    unsigned long Address;    ~511    /* 先頭アドレス */
    unsigned long Width;     ~512    /* 論理的縦幅 */
    unsigned long Height;    ~513    /* 論理的縦幅 */
    unsigned long a_Width;   ~514    /* 物理的実体の縦幅 */
    unsigned long a_Height;  ~515    /* 物理的実体の縦幅 */
    char depth;              ~516    /* 画素値の深さ */
) PTN_MCB;

```

(C)

```

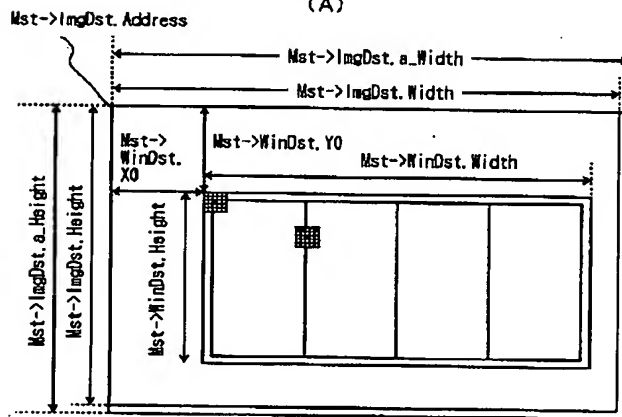
typedef struct(
    short x_coord;    ~521    /* 先頭のx座標 */
    short y_coord;    ~522    /* 先頭のy座標 */
    unsigned short Width;    ~523    /* ウィンドウの縦幅 */
    unsigned short Height;   ~524    /* ウィンドウの縦幅 */
) PTN_WCB;

```

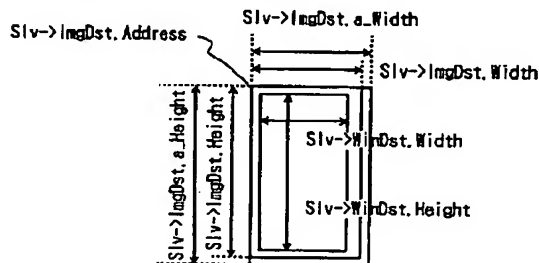
【図 8】

図 8

(A)



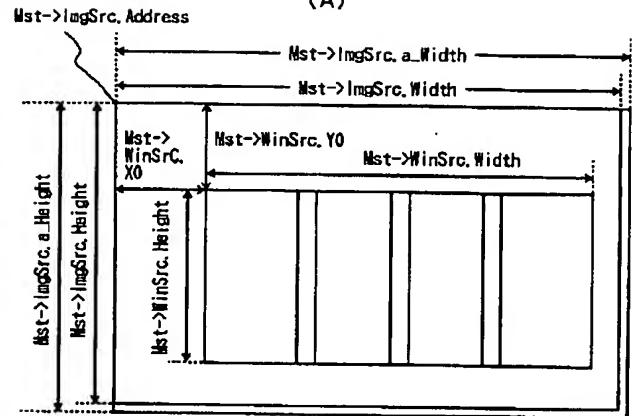
(B)



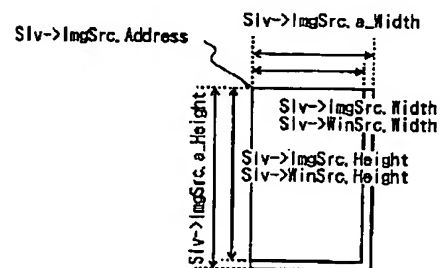
【図 7】

図 7

(A)



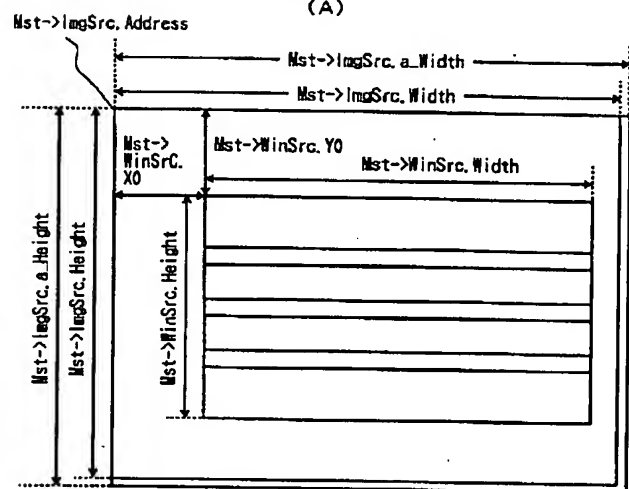
(B)



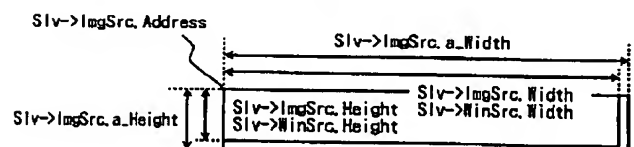
【図 9】

図 9

(A)



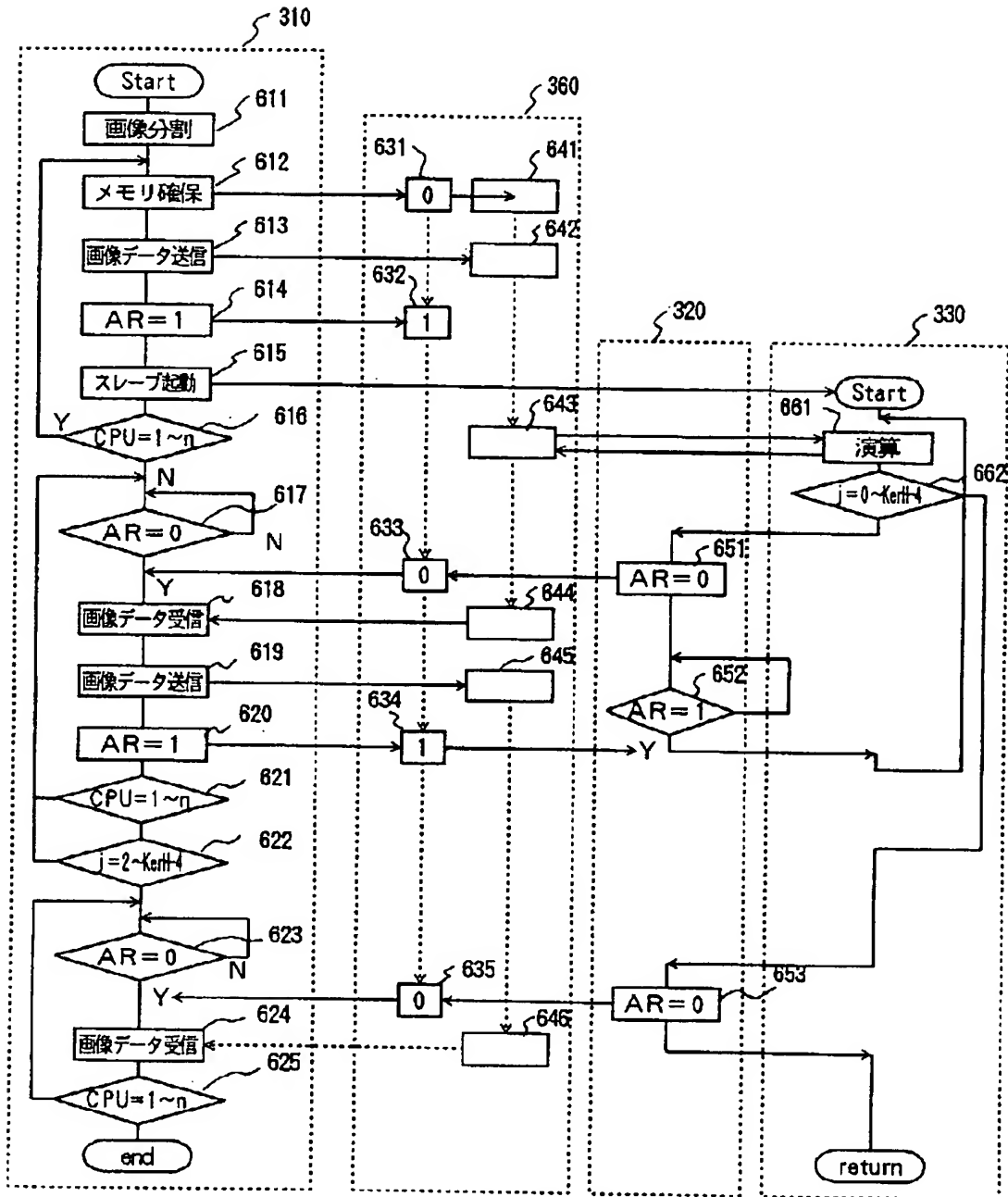
(B)



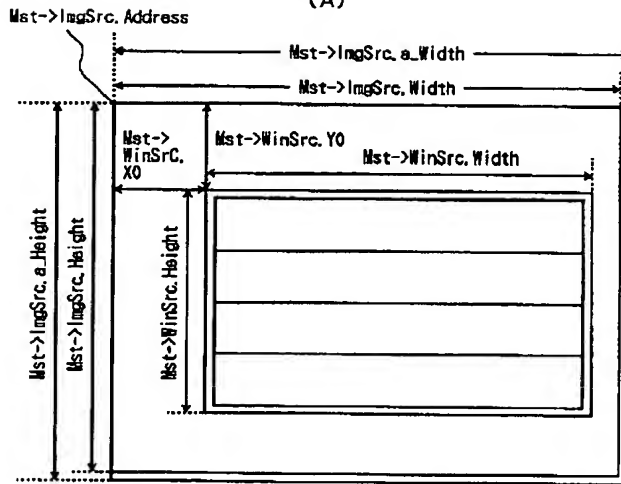


【図 6】

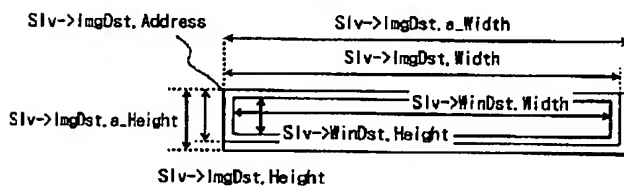
図 6



【図 10】

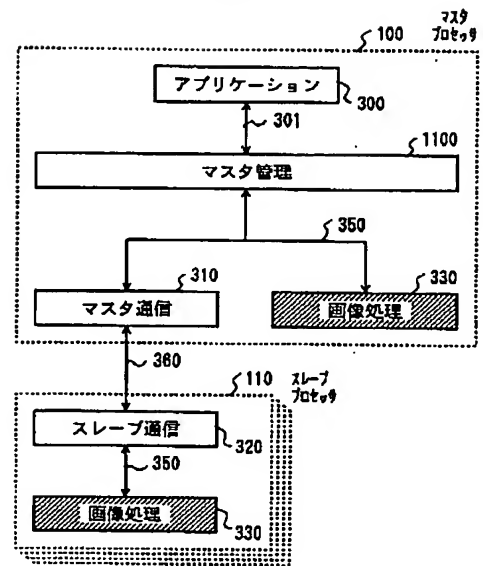
図 10  
(A)

(B)



【図 11】

図 11



フロントページの続き

(72) 発明者 高根 淳  
茨城県ひたちなか市大字市毛882番地 株  
式会社日立製作所計測器事業部内

(72) 発明者 佐藤 典夫  
茨城県ひたちなか市大字市毛882番地 株  
式会社日立製作所計測器事業部内  
Fターム(参考) 5B045 AA01 BB31 GG11  
5B057 CH04 CH14 CH16